LOVELY PROFESSIONAL UNIVERSITY

Transforming Education Transforming India

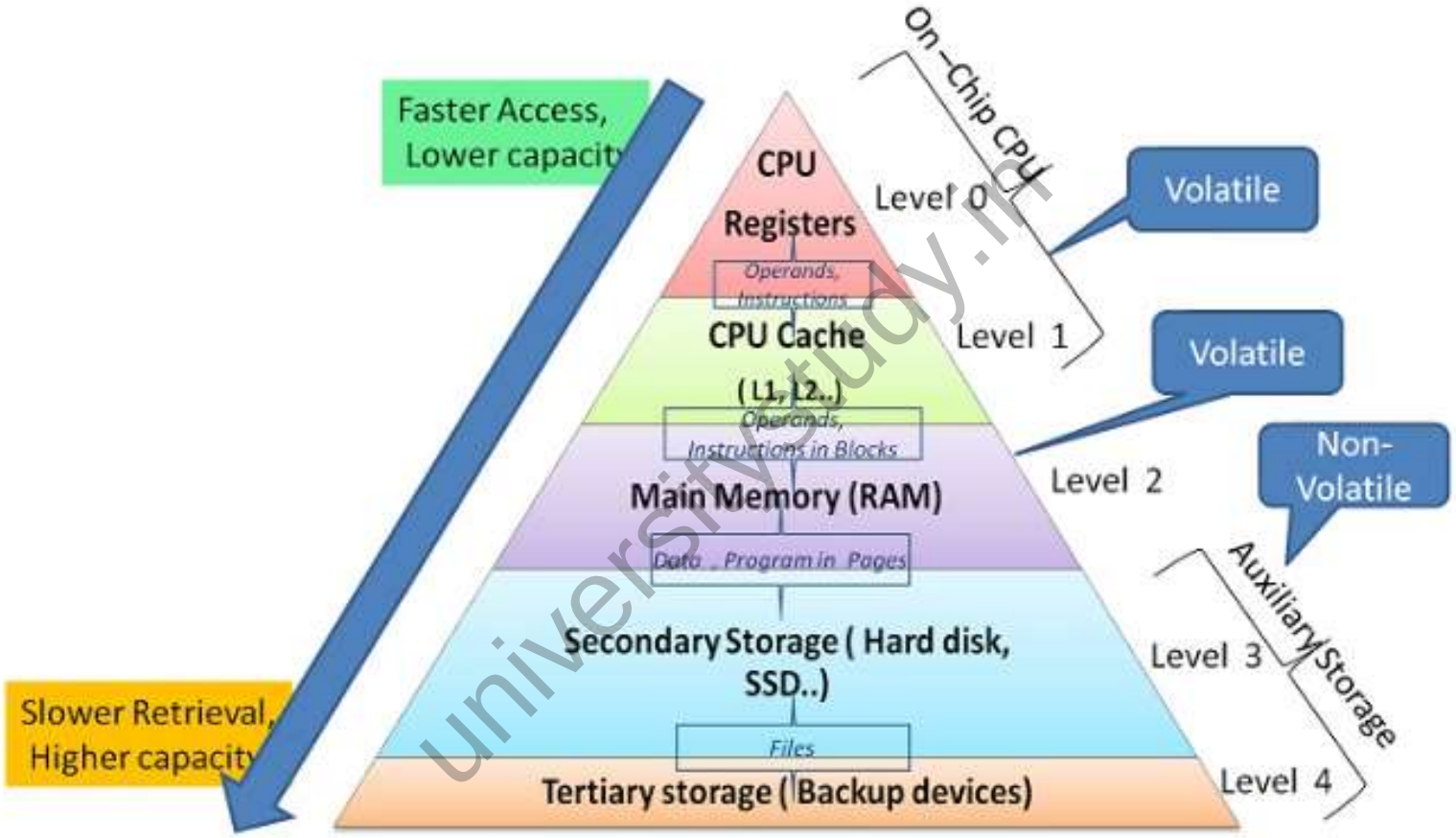
Orientation to Computing-I

L T P :2 0 0

Unit-1 (Computer Languages)

- Memory Hierarchy
- Machine Language
- Assembly Language
- High Level Language
- Steps in development of a Program
- Compilation and Execution
- Compiler
- Interpreter
- Assembler

Memory Hierarchy



Memory Hierarchy

Registers

- Registers are the fastest form of memory in a computer system. They are typically implemented using high-speed, low-latency hardware elements, such as flip-flops.
- These are the fastest and smallest storage units directly accessible by the CPU.
- Registers are directly accessible by the CPU's arithmetic and logic units (ALU) and other functional units, allowing for quick data manipulation.
- Registers hold data temporarily during program execution.

General Purpose Registers

- These registers are used for a wide range of data manipulation operations, including arithmetic, logic, and data storage.

- **EAX (Extended Accumulator):** EAX is often used as an accumulator register. It is designed for general-purpose arithmetic and logic operations like storing the results of arithmetic operations, holding return values from functions.

- **EBX (Extended Base Pointer)**: Serves as a base pointer when accessing data structures in memory, holds memory addresses or data.

- **ECX (Extended Counter)**: ECX is often used as a loop counter when iterating through loops, such as "for" or "while" loops and is used for performing string manipulation also.

Special Purpose Registers

- These registers are dedicated to specific functions within the CPU and play crucial roles in controlling and managing operations.

- **Program Counter (PC)**: Holds the memory address of the next instruction to be fetched and executed.

- **Stack Pointer (SP)**: Points to the top of the stack memory, used for managing function calls and data storage.

Cache

- Caches are small, high-speed memory units placed between the CPU and main memory (RAM). They store a subset of frequently used data and instructions to reduce the time it takes to access them.

Main Memory (RAM)

- Main memory is larger than cache memory but slower to access. It serves as the primary working memory for the CPU and stores both data and program instructions.

Secondary Storage

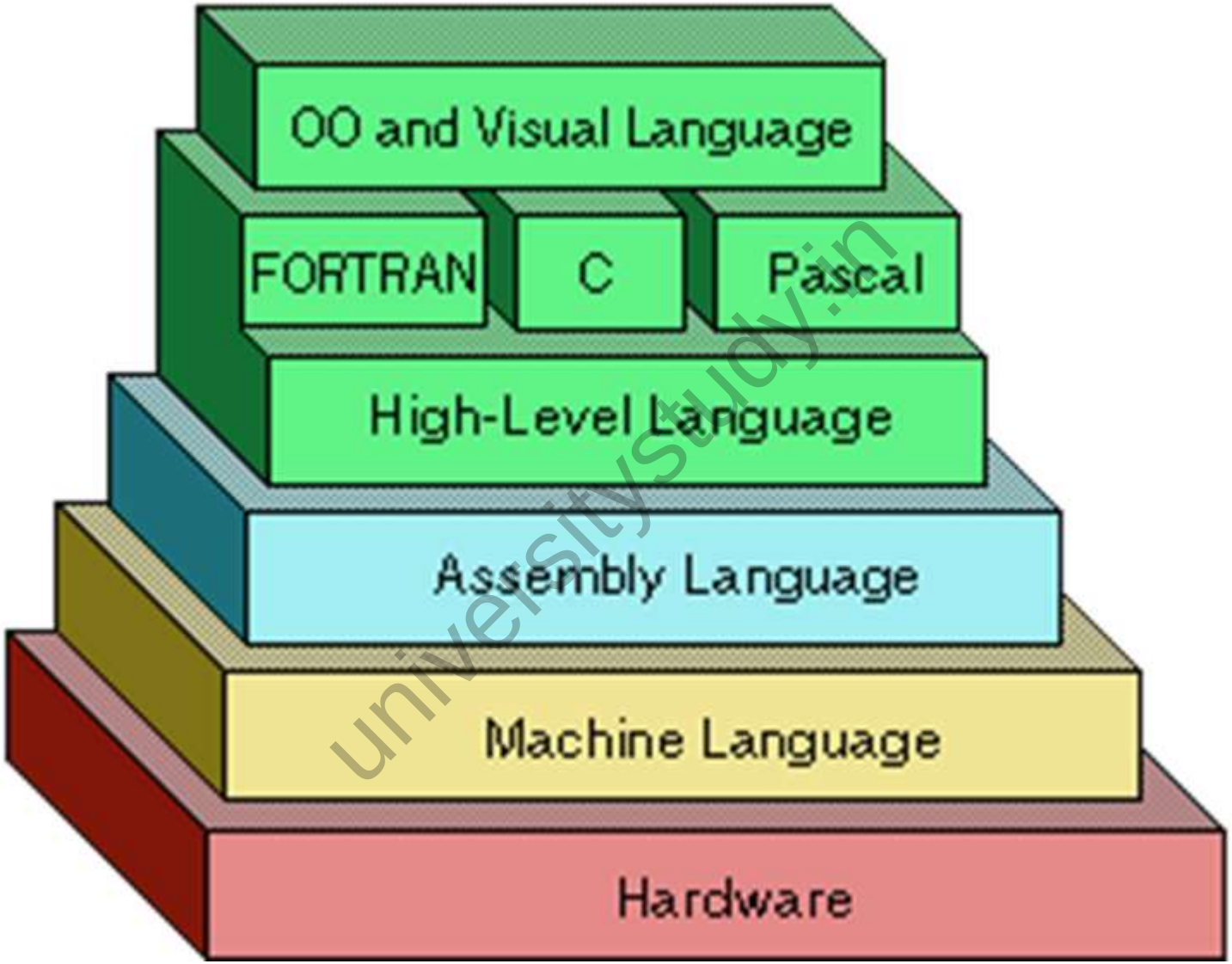
- This includes devices like hard drives, SSDs, and optical drives. They are much larger in capacity but slower than RAM. Secondary storage is used for long-term data storage and program storage.

Tertiary Storage

- Tertiary storage is commonly used for archival purposes, where data that is infrequently accessed but needs to be retained for a long time is stored. This includes data backups, historical records, and regulatory compliance data.

- Accessing data from tertiary storage is considerably slower compared to primary and secondary storage. Retrieving data from tapes, for example, involves physical movement of the tape and can take a relatively long time.

- Tertiary storage has evolved over time, and newer technologies, such as cloud storage and high-capacity optical discs, have provided alternatives to traditional magnetic tape-based tertiary storage.



MACHINE LANGUAGE

- Machine language is a low-level language made up of binary numbers or bits that a computer can understand.
- It is also known as machine code or object code and is extremely tough to comprehend.
- The only language that the computer understands is machine language. All programmes and programming languages, such as Swift and C++, produce or run programmes in machine language before they are run on a computer.
- When a specific task, even the smallest process executes, machine language is transported to the system processor. Computers are only able to understand binary data as they are digital devices.

MACHINE LANGUAGE

- Machine languages consist of strings of numbers ultimately reduced to 1's and 0's. The computer responds to these numbers by performing different operations.

- Each computer can directly understand only one language—its own machine language. With machine language a programmer can instruct a computer to perform its most fundamental operations.

- Programs written in machine language are not portable, that is, they may not be run on other computers with different machine languages.

- Machine language programs are easy for computers to understand, but, for people, machine language programming is tedious work susceptible to error.

ASSEMBLY LANGUAGE

- In computer programming, assembly language (or assembler language, or symbolic machine code), is any low-level programming language with a very strong correspondence between the instructions in the language and the architecture's machine code instructions.

What does assembly code do?

- Assembly code is converted into executable machine code by a utility program referred to as an assembler.
- The conversion process is referred to as assembly, as in assembling the source code. The computational step when an assembler is processing a program is called assembly time.
- Assembly code is generally used for low-level programming, debugging, or specialized tasks rather than as an intermediate step in regular software development.

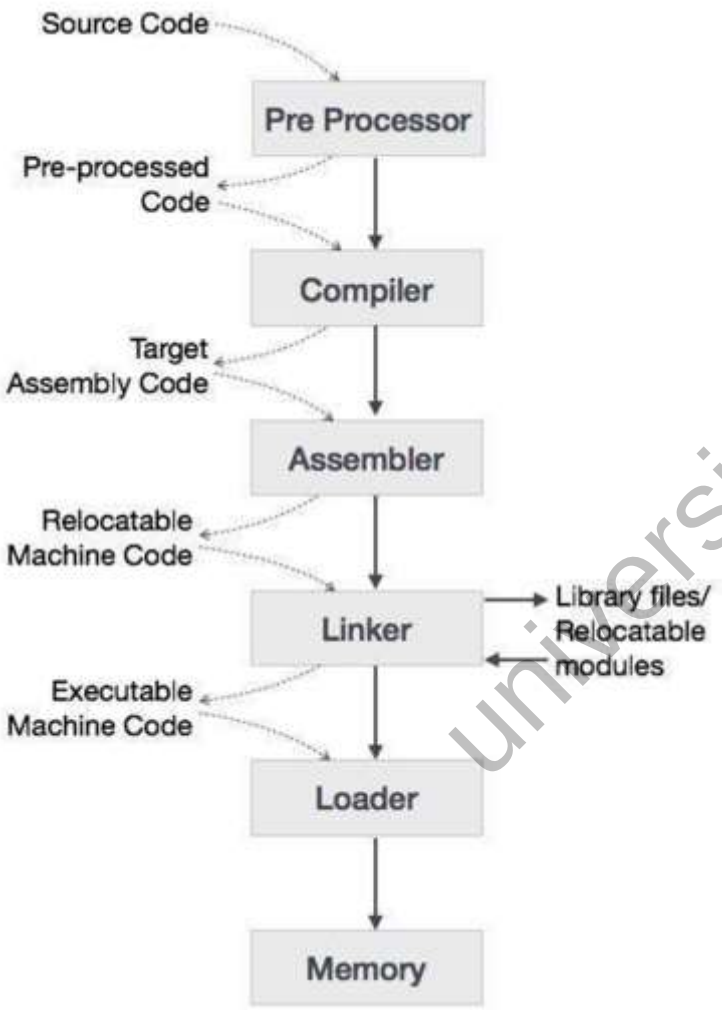
Parameters	Machine Language	Assembly Language
Nature of Syntax	The machine languages consist of 1s and 0s as binary digits.	The assembly languages have a similar syntax to that of the English language- thus making it easy for general users to understand (other than the programmers).
Ease of Comprehension	Only computers can comprehend machine languages. A normal human doesn't possess the capacity to decipher it.	It is very easy for any human to understand, apply, and use assembly language. Machines cannot directly *read* and *understand* them.
Level of Language	The machine language is the lowest level of all the programming languages. It executes all the instructions directly through the CPU (Central Processing System) of the system.	The assembly language is a low-level language for programming that requires an assembler to convert the instructions into a final object or machine code.
Dependency	The machine languages stay dependent on the concerned platforms. Their features have a variation accordingly.	The assembly languages consist of a standard set of instructions.
Applicable Areas	These languages only serve in the case of coding for machines.	One can use these languages in microprocessor-based devices/ apps and also for real-time systems.

High Level Languages

- A HIGH-LEVEL LANGUAGE (HLL) is A programming language such as C, FORTRAN, or pascal that enables a programmer to write programs that are more or less independent of a particular type of computer.

- Such languages are considered high-level because they are closer to human languages and further from machine languages

Steps in development



LOOK AT THIS FLOW

The high-level language is converted into binary language in various phases. A compiler is a program that converts high-level language to assembly language. Similarly, an assembler is a program that converts the assembly language to machine-level language.

Let us first understand how a program, using C compiler, is executed on a host machine.

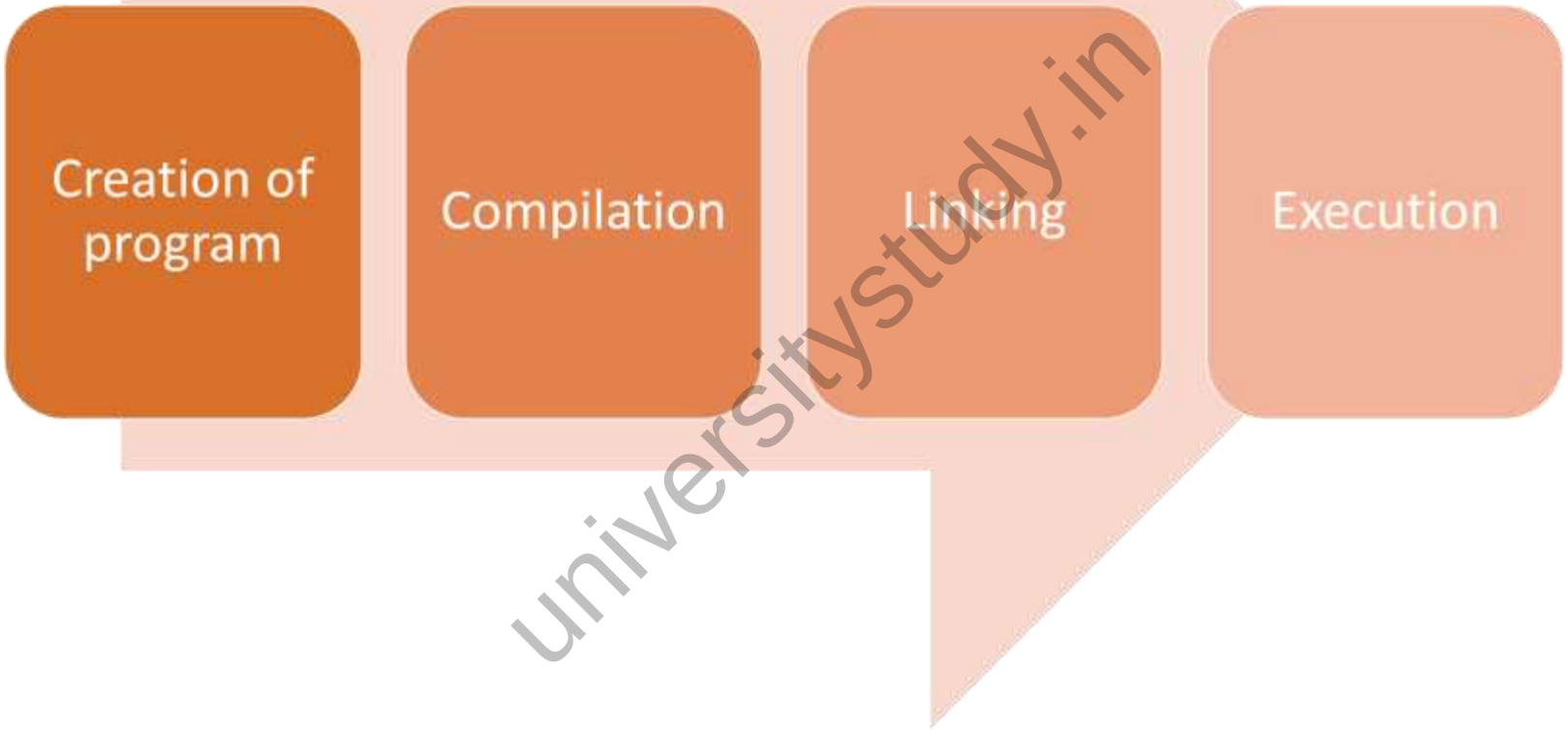
Before diving straight into the concepts of compilers, we should understand a few other tools that work closely with compilers.

Preprocessor

A preprocessor, generally considered as a part of compiler, is a tool that produces input for compilers. It deals with macro-processing, augmentation, file inclusion, language extension, etc.

Work Flow

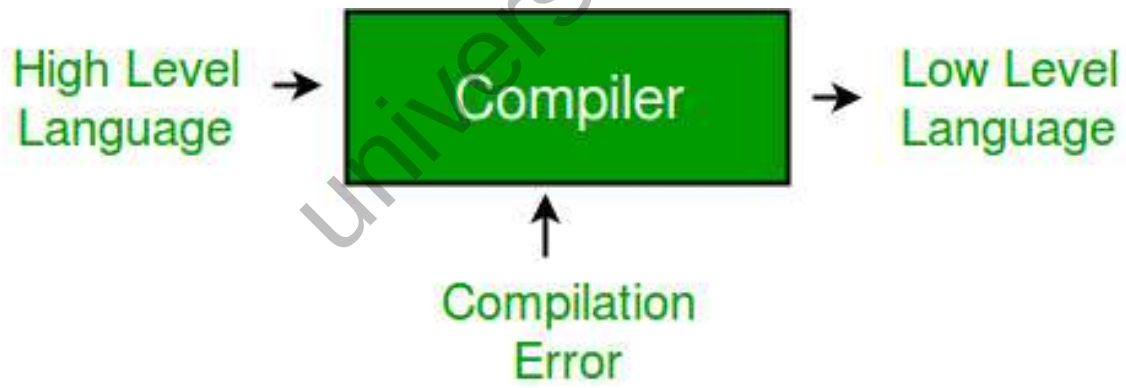
- USER WRITES A PROGRAM IN C LANGUAGE (HIGH-LEVEL LANGUAGE).
- THE C COMPILER, COMPILES THE PROGRAM AND TRANSLATES IT TO ASSEMBLY PROGRAM (LOW-LEVEL LANGUAGE).
- AN ASSEMBLER THEN TRANSLATES THE ASSEMBLY PROGRAM INTO MACHINE CODE (OBJECT).
- A LINKER TOOL IS USED TO LINK ALL THE PARTS OF THE PROGRAM TOGETHER FOR EXECUTION (EXECUTABLE MACHINE CODE).
- A LOADER LOADS ALL OF THEM INTO MEMORY AND THEN THE PROGRAM IS EXECUTED.



Compiler

- A compiler is a special program that translates a programming language's source code into machine code, bytecode or another programming language.

- The source code is typically written in a high-level, human-readable language such as Java or C++.



Interpreter

- An interpreter, like a compiler, translates high-level language into low-level machine language. The difference lies in the way they read the source code or input.
- A compiler reads the whole source code at once, creates tokens, checks semantics, generates intermediate code, executes the whole program and may involve many passes.
- In contrast, an interpreter reads a statement from the input, converts it to an intermediate code, executes it, then takes the next statement in sequence. If an error occurs, an interpreter stops execution and reports it. whereas a compiler reads the whole program even if it encounters several errors.

Assembler

- An assembler translates assembly language programs into machine code.
- The output of an assembler is called an object file, which contains a combination of machine instructions as well as the data required to place these instructions in memory.
- An assembler is a type of computer program that interprets software programs written in assembly language into machine language, code and instructions that can be executed by a computer.
- An assembler enables software and application developers to access, operate and manage a computer's hardware architecture and components